在 iOS 上用 Core Image 实现人脸检测

原创 2017-05-11 Gregg SwiftGG翻译组

> 译者：智多芯；校对：Crystal Sun；定稿：CMB

Core Image 是 Cocoa Touch 框架提供的功能强大的 API，是 iOS SDK 中常常被忽视的关键部件。本教程将尝试探索 Core Image 提供的人脸识别功能，并将其应用到 iOS App 中。

> 注：这是中高级 iOS 教程，本教程假设你已经使用过类似 UIImagePicker，Core Image 等技术。如果你对这些还不熟悉，先看看我们的 iOS 教程系列，等你准备好了再看这篇文章。

接下来要做的事

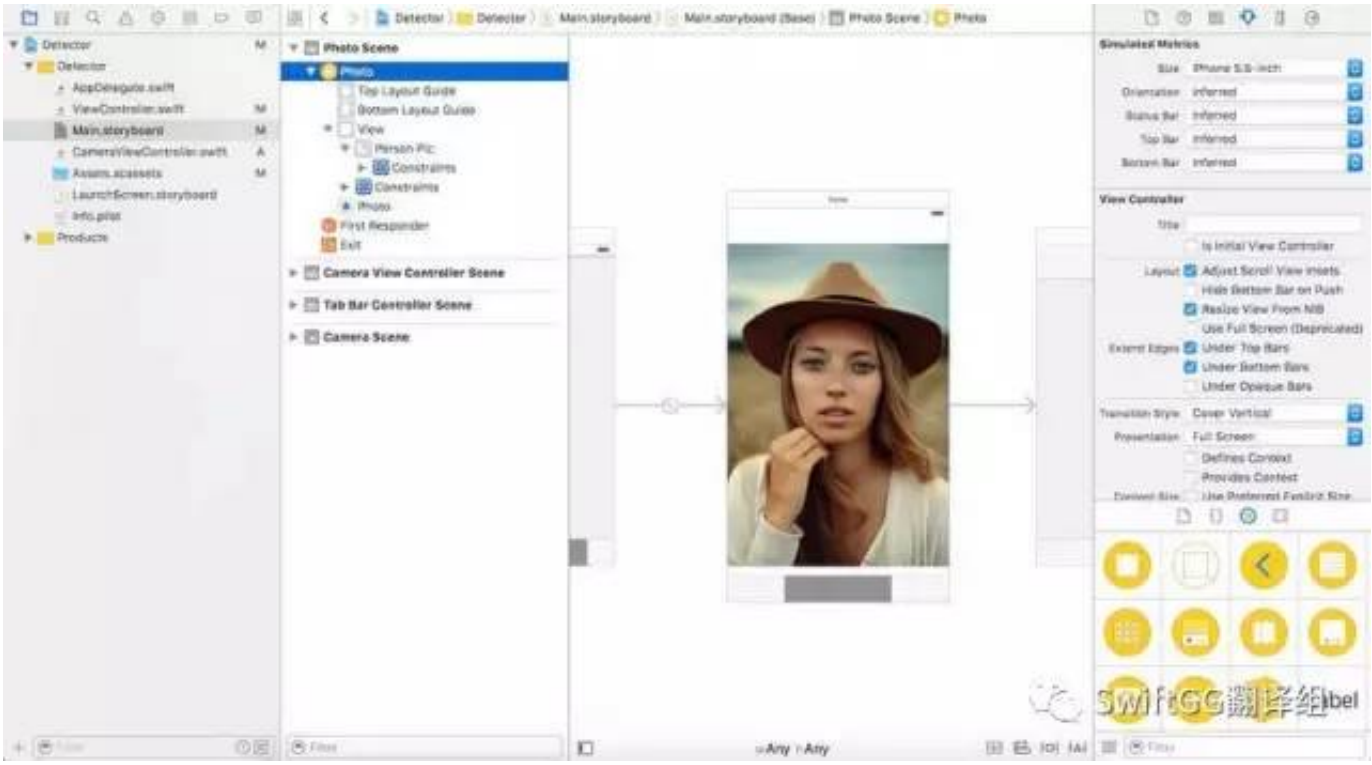
自从 iOS 5（大概在2011年左右）之后，iOS 开始支持人脸识别，只是用的人不多。人脸识别 API 让开发者不仅可以进行人脸检测，还能识别微笑、眨眼等表情。

首先创建一个简单的应用，探索一下 Core Image 提供的人脸识别技术，该应用可以识别出照片中的人脸并用方框将人脸框起来。在第二个例子中，用户可以拍照并检测照片上是否有人脸出现，如果有则提取人脸坐标。通过这两个例子，你将学会 iOS 上所有关于人脸识别的技术，并充分利用它强大却经常被忽视的功能。

下面开始吧！

设置工程

下载并在 Xcode 中打开起始工程。该工程中的 Storyboard 仅包含一个已连接到代码的 IBOutlet 和 imageView。



> 注：项目中的图片由 unsplash.com 提供。

在开始使用 Core Image 进行人脸识别之前，需要将 Core Image 库导入项目中。打开 ViewController.swift 文件，在文件最上方插入如下代码：

```
import CoreImage
```

用 Core Image 实现人脸检测

在起始工程的 storyboard 里包含一个通过 IBOutlet 连接到代码中的 imageView。下一步将实现人脸检测的代码。先把以下代码加入 swift 文件中，后面再解释：

```
func detect() {
    guard let personciImage = CIImage(image: personPic.image!) else {
        return
    }

    let accuracy = [CIDetectorAccuracy: CIDetectorAccuracyHigh]
    let faceDetector = CIDetector(ofType: CIDetectorTypeFace, context: nil,
options: accuracy)
    let faces = faceDetector?.features(in: personciImage)

    for face in faces as! [CIFaceFeature] {
```

```
        print("Found bounds are \(face.bounds)")

        let faceBox = UIView(frame: face.bounds)
        faceBox.layer.borderWidth = 3
        faceBox.layer.borderColor = UIColor.red.cgColor
        faceBox.backgroundColor = UIColor.clear
        personPic.addSubview(faceBox)

        if face.hasLeftEyePosition {
            print("Left eye bounds are \(face.leftEyePosition)")
        }

        if face.hasRightEyePosition {
            print("Right eye bounds are \(face.rightEyePosition)")
        }
    }
}
```

这里解释一下上面的代码：

- 第 3 行：从 storyboard 中的 UIImageView 提取出 UIImage 并转换成 CIImage，将其保存在新创建的 `personciImage` 变量中。Core Image 需要用到 CIImage。
- 第 7 行：创建一个 `accuracy` 变量并设置为 `CIDetectorAccuracyHigh`。你可以选择 `CIDetectorAccuracyHigh` 或 `CIDetectorAccuracyLow`。本文希望得到高精度的结果，因此选择了 `CIDetectorAccuracyHigh`。
- 第 8 行：创建一个 `faceDetector` 变量并设置为 `CIDetector` 的实例。实例化 `CIDetector` 时将前文创建的 `accuracy` 作为参数传入。
- 第 9 行：通过调用 `faceDetector` 的 `features(in:)` 方法可检测出给定图像的所有人脸，最终以数组的形式返回所有人脸。
- 第 11 行：遍历数组中所有的人脸，并将其转换为 `CIFaceFeature` 类型。
- 第 15 行：创建一个 UIView 实例并命名为 `faceBox`，然后根据 `faces.first` 设置其大小。这将画一个方框用于高亮检测到的人脸。
- 第 17 行：将 `faceBox` 的边框宽度设为 3。
- 第 18 行：将边框颜色设置为红色。
- 第 19 行：将背景色设为透明，表示该视图没有可见的背景。
- 第 20 行：最后，将该视图添加到 `personPic` 视图中。
- 第 22-28 行：这些 API 不仅可以检测出人脸，还能检测出人脸的左右眼，但本文就不在图像中高亮人眼了。本文只想展示一些 `CIFaceFeature` 的相关属性。

接着调用在 `viewDidLoad` 中调用 `detect` 方法，在方法中增加下列一行代码：

```
detect()
```

编译并运行程序，可以看到如下效果：

根据控制台的输出结果，似乎可以检测出人脸：

```
Found bounds are (177.0, 415.0, 380.0, 380.0)
```

还有几个问题没有处理：

- 人脸识别程序应用于原始图像上，而原始图像有着比 imageView 更高的分辨率。另外，工程中 imageView 的 content mode 被设置为 aspect fit。为了正确地画出检测框，还需要计算出 imageView 中识别到的人脸的实际位置和尺寸。
- 再者，Core Image 和 UIView（或者UIKit）使用了不同的坐标系（如下图所示），因此还

需要实现 Core Image 坐标到 UIView 坐标的转换。

现在使用下面的代码替换 `detect()` 方法中的代码：

```
func detect() {
    guard let personciImage = CIImage(image: personPic.image!) else {
        return
    }

    let accuracy = [CIDetectorAccuracy: CIDetectorAccuracyHigh]
    let faceDetector = CIDetector(ofType: CIDetectorTypeFace, context: nil,
options: accuracy)
    let faces = faceDetector?.features(in: personciImage)

    // 将 Core Image 坐标转换成 UIView 坐标
    let ciImageSize = personciImage.extent.size
    var transform = CGAffineTransform(scaleX: 1, y: -1)
    transform = transform.translatedBy(x: 0, y: -ciImageSize.height)

    for face in faces as! [CIFaceFeature] {
        print("Found bounds are \(face.bounds)")

        // 实现坐标转换
```

```
        var faceViewBounds = face.bounds.applying(transform)

        // 计算实际的位置和大小
        let viewSize = personPic.bounds.size
        let scale = min(viewSize.width / ciImageSize.width,
                        viewSize.height / ciImageSize.height)
        let offsetX = (viewSize.width - ciImageSize.width * scale) / 2
        let offsetY = (viewSize.height - ciImageSize.height * scale) / 2

        faceViewBounds = faceViewBounds.applying(CGAffineTransform(scaleX: s
cale, y: scale))
        faceViewBounds.origin.x += offsetX
        faceViewBounds.origin.y += offsetY

        let faceBox = UIView(frame: faceViewBounds)

        faceBox.layer.borderWidth = 3
        faceBox.layer.borderColor = UIColor.red.cgColor
        faceBox.backgroundColor = UIColor.clear
        personPic.addSubview(faceBox)

        if face.hasLeftEyePosition {
            print("Left eye bounds are \(face.leftEyePosition)")
        }

        if face.hasRightEyePosition {
            print("Right eye bounds are \(face.rightEyePosition)")
        }
    }
}
```

首先，上面的代码使用放射变换将 Core Image 坐标转换成了 UIKit 坐标。然后，添加了一些额外的代码用于计算框视图的实际位置和尺寸。

现在再一次运行程序，应该可以看到检测框将识别出的人脸框起来了，这样就成功地用 Core Image 检测到人脸了。

开发一个支持人脸识别的摄像应用

假设有一个用于摄像或拍照的应用程序，我们希望在拍照后检测是否有人脸出现。如果出现了人脸，可能想将这张照片打上一些标签并对其分类。下面结合 `UIImagePicker` 类，拍照完成时立刻运行上面的人脸检测代码。

上面的起始工程中已经创建了一个 `CameraViewController` 类，将其代码更新成下面这样，用以实现摄像功能：

```
class CameraViewController: UIViewController, UIImagePickerControllerDelegat
e, UINavigationControllerDelegate {
    @IBOutlet var imageView: UIImageView!
    let imagePicker = UIImagePickerController()

    override func viewDidLoad() {
        super.viewDidLoad()

        imagePicker.delegate = self
    }
```

```
    @IBAction func takePhoto(sender: AnyObject) {
        if !UIImagePickerController.isSourceTypeAvailable(.camera) {
            return
        }

        imagePicker.allowsEditing = false
        imagePicker.sourceType = .camera

        present(imagePicker, animated: true, completion: nil)
    }

    func imagePickerController(_ picker: UIImagePickerController, didFinishP
ickingMediaWithInfo info: [String : Any]) {
        if let pickedImage = info[UIImagePickerControllerOriginalImage] as?
UIImage {
            imageView.contentMode = .scaleAspectFit
            imageView.image = pickedImage
        }

        dismiss(animated: true, completion: nil)
        self.detect()
    }

    func imagePickerControllerDidCancel(_ picker: UIImagePickerController) {
        dismiss(animated: true, completion: nil)
    }
}
```

开始的几行代码设置了 `UIImagePicker` 代理。在 `didFinishPickingMediaWithInfo` 方法（这是一个 `UIImagePicker` 代理方法）中，将传入的图像设置到 imageView 上，最后关闭拾取器并调用 `detect` 函数。

上面的代码还未实现 `detect` 函数，将下面的代码加上：

```
func detect() {
    let imageOptions = NSDictionary(object: NSNumber(value: 5) as NSNumber,
forKey: CIDetectorImageOrientation as NSString)
    let personciImage = CIImage(cgImage: imageView.image!.cgImage!)
    let accuracy = [CIDetectorAccuracy: CIDetectorAccuracyHigh]
    let faceDetector = CIDetector(ofType: CIDetectorTypeFace, context: nil,
options: accuracy)
    let faces = faceDetector?.features(in: personciImage, options: imageOpti
ons as? [String : AnyObject])

    if let face = faces?.first as? CIFaceFeature {
        print("found bounds are \(face.bounds)")
```

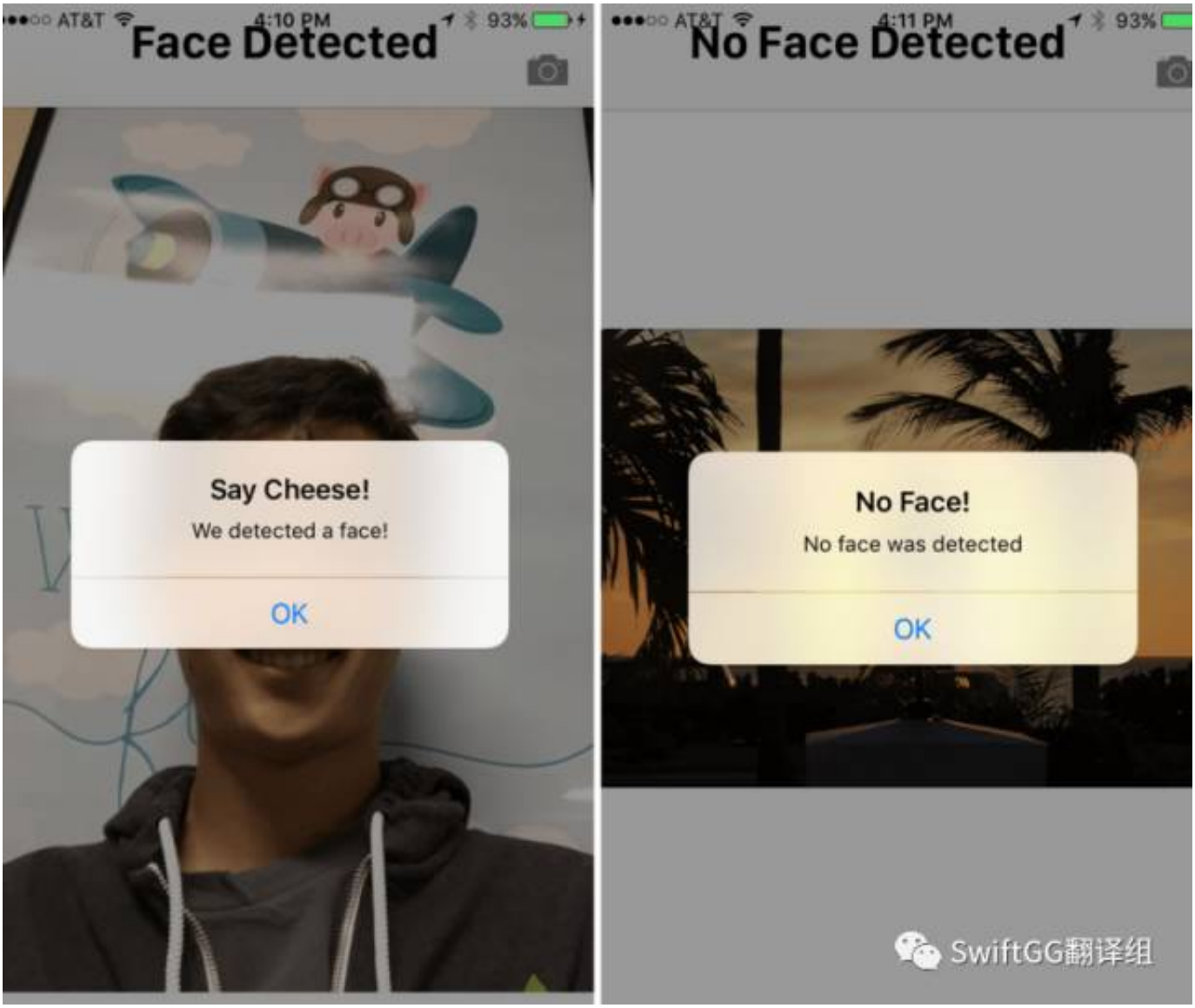
```
        let alert = UIAlertController(title: "Say Cheese!", message: "We det
ected a face!", preferredStyle: UIAlertControllerStyle.alert)
        alert.addAction(UIAlertAction(title: "OK", style: UIAlertActionStyle
.default, handler: nil))
        self.present(alert, animated: true, completion: nil)

        if face.hasSmile {
            print("face is smiling");
        }

        if face.hasLeftEyePosition {
            print("Left eye bounds are \(face.leftEyePosition)")
        }

        if face.hasRightEyePosition {
            print("Right eye bounds are \(face.rightEyePosition)")
        }
    } else {
        let alert = UIAlertController(title: "No Face!", message: "No face w
as detected", preferredStyle: UIAlertControllerStyle.alert)
        alert.addAction(UIAlertAction(title: "OK", style: UIAlertActionStyle
.default, handler: nil))
        self.present(alert, animated: true, completion: nil)
    }
}
```

这里的 detect() 函数和之前的实现非常相似，不过这一次我们使用的是临时拍到的图像。根据检测结果会显示一个提示框，提示是否检测到人脸。运行程序来快速测试一下。



CIFaceFeature 中的一些属性和方法前面已经尝试过了。例如，若要判断照片中的人是否正在微笑，可以通过 `hasSmile` 属性判断。还可以通过 `hasLeftEyePosition`（或 `hasRightEyePosition`）属性检查是否有左眼（或右眼）出现（希望有）。

还可以通过 `hasMouthPosition` 来判断是否出现了嘴巴。如果出现了，可以通过 `mouthPosition` 属性得到其坐标，代码如下：

```
if (face.hasMouthPosition) {
     print("mouth detected")
}
```

如你所见，通过 Core Image 进行人脸识别极其简单。除了检测嘴、微笑、眼睛位置等，还可以通过 `leftEyeClosed`（或 `rightEyeClosed`）判断左眼（或右眼）是否睁开。

结语

本教程探索了 Core Image 提供的人脸识别 API，并展示了如何在摄像机应用中使用该功能。本文通过 UIImagePicker 拍摄图像，并检测该图像中是否有人的出现。

如你所见，Core Image 的人脸识别 API 有着非常多的用处！希望你能觉得本教程有所帮助，让你了解到了这一鲜为人知的 iOS API！

> 注：欢迎继续关注让人脸识别更加强大的神经网络系列教程。

你可以从这里下载到最终的工程代码。

> 本文由 SwiftGG 翻译组翻译，已经获得作者翻译授权，最新文章请访问 http://swift.gg。

阅读原文
